Modified orbital branching for structured symmetry with an application to unit commitment

Mathematical Programming Series A manuscript No.
(will be inserted by the editor)

Modified Orbital Branching for Structured Symmetry with an Application to Unit Commitment

James Ostrowski · Miguel F. Anjos · Anthony Vannelli

Received: date / Accepted: date

Abstract The past decade has seen advances in general methods for symmetry breaking in mixed-integer linear programming. These methods are advantageous for general problems with general symmetry groups. Some important classes of mixed integer linear programming problems, such as bin packing and graph coloring, contain highly structured symmetry groups. This observation has motivated the development of problem-specific techniques. In this paper we show how to strengthen orbital branching in order to exploit special structures in a problem's symmetry group. The proposed technique, to which we refer as modified orbital branching, is able to solve problems with structured symmetry groups more efficiently. One class of problems for which this technique is effective is when the solution variables can be expressed as 0/1 matrices where the problem's symmetry group contains all permutations of the columns. We use the unit commitment problem, an important problem in power systems, to demonstrate the strength of modified orbital branching.

Keywords symmetry · integer programming · orbital branching · orbitopes · unit commitment

James Ostrowski
Department of Industrial and Systems Engineering
University of Tennessee Knoxville
Knoxville, TN, United States
E-mail: jostrows@utk.edu

Miguel F. Anjos
Canada Research Chair in Discrete Nonlinear Optimization in Engineering
GERAD & École Polytechnique de Montréal
Montréal, QC, Canada H3C 3A7
E-mail: anjos@stanfordalumni.org

Anthony Vannelli
School of Engineering
University of Guelph
Guelph, ON, Canada N1G 2W1
E-mail: vannelli@uoguelph.ca

1 Introduction

The presence of symmetry in mixed integer linear programming (MILP) has long caused computational difficulties. Jeroslow [9] introduced a class of simple integer programming problems that require an exponential number of subproblems when using pure branch and bound due to the problem's large symmetry group. The past decade has seen advances in general methods for symmetry breaking, most notably isomorphism pruning [14,15] and orbital branching [18]. These methods are advantageous for general problems with general symmetry groups. There are important classes of MILP problems, such as bin packing and graph coloring, that contain highly structured symmetry groups which can be exploited. This observation has motivated the development of problem-specific techniques. For example, orbitopal fixing [11,12] is an efficient way to break symmetry in bin packing problems. Symmetry breaking constraints can be added to formulations of telecommunication problems, noise pollution problems, and others, see e.g. [20].

In some cases, a general symmetry breaking-method such as orbital branching can be modified to take advantage of special structure. For example, consider the Jeroslow problem

$$\begin{array}{ll} \min & x_{n+1} \\ \text{s.t.} & \sum_{i=1}^{n} 2x_i + x_{n+1} = 2\lfloor \frac{n}{2} \rfloor + 1 \\ & x_i \in \{0,1\} \ \forall i \in \{1,\ldots,\ n+1\}, \end{array} \tag{1}$$

for any positive integer n. This problem contains a large amount of symmetry. Using pure branch-and-bound, solving (1) will require an exponentially large branch-and-bound tree [9]. However, this symmetry is very well structured. Any permutation of variables among $\{x_1,\ldots,\ x_n\}$, while leaving the remaining variables unchanged, is a symmetry. The only permutations that are not symmetries are those that move the variable x_{n+1}. Because of this special structure, the constraints $x_i \geq x_{i+1}$, for $i \in \{1,\ldots,n-1\}$, can be added to the problem to break the symmetry. With these constraints, a pure branch-and-bound approach can solve (1) using just one branch. Similarly, the problem can be reformulated by replacing $\sum_{i=1}^{n} x_i$ by y; the resulting formulation is easy to solve. General symmetry-breaking methods are less efficient. Both isomorphism pruning [14,15] and orbital branching [18] need $\lfloor \frac{n}{2} \rfloor + 1$ nodes to solve (1).

In this paper we show how to strengthen orbital branching in cases where the problem's symmetry group contains additional structure. The proposed technique, to which we refer as modified orbital branching, is able to solve problems with structured symmetry groups more efficiently. For example, modified orbital branching solves (1) in one branch. This improvement is especially useful for problems whose solutions can be expressed as orbitopes, e.g. like those studied in [11,12]. The proposed modified orbital branching strengthens the results in [11,12] by extending the classes of problems for which symmetry can be efficiently removed.

We demonstrate the potential of modified orbital branching for problems with structured symmetry via the unit commitment (UC) problem. The UC problem is formulated in the context of operating an electrical power system and consists of finding an optimal power production schedule for each generator in the system so as to minimize the total cost of power generation while ensuring that demand is met and that the system operates safely and reliably. Real-world instances of UC lead to challenging optimization problems, in particular in the presence of multiple generators with identical characteristics since this results in problems with significant symmetrical structure. The survey paper [1] provides an introduction to UC from the point of view of optimization and a summary of the recent developments in the literature.

This paper is structured as follows. Section 2 provides an overview of symmetry in integer programming, as well as a description of orbital branching. Section 3 shows how orbital branching can be strengthened when the problem's symmetry group has a special structure. Section 4 shows how modified orbital branching can be used to efficiently remove all isomorphic solutions from MILP problems that can be expressed as orbitopes. Section 5 demonstrates the strength of modified orbital branching on UC problems having multiple generators with the same characteristics. Section 6 concludes the paper.

2 Symmetry and Orbital Branching

2.1 Background on Symmetry

The set S^n is the set of all permutations of $I^n = \{1, \ldots, n\}$. This set forms the *full symmetric group* of I^n. Any subgroup of the full symmetric group is a *permutation group*. For any permutation group Γ, the following hold:

- Γ contains the *identity* permutation e.
- If $\pi \in \Gamma$, then $\pi^{-1} \in \Gamma$, where π^{-1} is the *inverse* permutation.
- For π_1 and $\pi_2 \in \Gamma$, the *composition* of π_1 and π_2, denoted $\pi_1 \circ \pi_2$, is in Γ.

The permutation $\pi \in \Gamma$ maps the point $z \in \mathbb{R}^n$ to $\pi(z)$ by permuting the indices. The permutation group Γ acts on a set of points $\mathcal{Z} \subset \mathbb{R}^n$ if $\Gamma(\mathcal{Z}) = \mathcal{Z}$ holds.

We focus on MILPs of the form $\min\{c^T x + d^T y \mid Ax + By \leq b,\ x \in \mathbb{Z}^n_+,\ y \in R^p_+\}$, where c is an n-vector, d a p-vector, A an $m \times n$ matrix, B an $m \times p$ matrix, and b an m-vector. Let $\mathcal{F}$ denote the feasible set of a given MILP. The *symmetry group* $\mathcal{G}$ of an MILP is the set of permutations of the variables that map each feasible solution onto a feasible solution of the same value:

$$\mathcal{G} \stackrel{\text{def}}{=} \{\pi \in S^n \mid \pi(x) \in \mathcal{F} \text{ and } c^T x = c^T \pi(x) \quad \forall x \in \mathcal{F}\}.$$

Note that computing a problem's symmetry group is NP-hard [16].

The *orbit* of z under the action of the group $\mathcal{G}$ is the set of all elements of $\mathcal{Z}$ to which z can be sent by permutations in $\mathcal{G}$:

$$\text{orb}(z, \mathcal{G}) \stackrel{\text{def}}{=} \{\pi(z) \mid \pi \in \mathcal{G}\}.$$

Orbits can also be extended to variables. We say that $\{x_{i_1},\ x_{i_2}, \ldots,\ x_{i_k}\}$ is a (variable) orbit whenever $\text{orb}(\{e_{i_1}\}, \mathcal{G}) = \{e_{i_1},\ e_{i_2}, \ldots,\ e_{i_k}\}$. Therefore, the union of the orbits

$$\bigcup_{j=1}^{n} \text{orb}(\{e_j\}, \mathcal{G}) \tag{2}$$

can be interpreted as a partition of I^n, as well as the variables.

To study how a group Γ acts on a subset of I^n, we *project* Γ. The projection of Γ on $J \subseteq I^n$, $\texttt{Proj}_J(\Gamma)$, is formed by truncating the actions of Γ onto the set J and choosing the set of permutations $\pi_P : J \to J$ such that

$$\pi_P \in \texttt{Proj}_J(\Gamma) \Leftrightarrow \exists \pi \in \Gamma \text{ s.t. } \pi(e_i) = \pi_P(e_i)\ \forall\ i \in J. \tag{3}$$

Note that it only makes sense to project the symmetry group Γ onto a set J if for any $i \in J$, k must be in J if there exists a $\pi \in \Gamma$ with $e_k = \pi(e_i)$.

If Γ is the set of all permutations of pairwise distinct elements $\{i_1, \ldots,\ i_n\}$, then we say that Γ is isomorphic to S^n, or $\Gamma \cong S^n$. If J represents an orbit of e_i and $\texttt{Proj}_J(\Gamma)$ consists of all permutations of the elements of the orbit, then $\texttt{Proj}_J(\Gamma) \cong S^{|J|}$.

Example 1 Permutations are commonly written in *cycle notation*. The expression $(a_1,\ a_2, \ldots a_k)$ denotes a cycle which sends the entry at index a_i to index a_{i+1} for $i = 1, \ldots,\ k-1$ and sends the entry at index a_k to index a_1. Permutations can be written as a product of cycles. For example, the expression $(a_1,\ a_2)(a_3)$ denotes a permutation which sends the entry at index a_1 to a_2, a_2 to a_1, and a_3 to itself. We omit 1-element cycles for clarity.

Let $\Gamma \subset S^5$ be the permutation group containing the following permutations:

$$\begin{aligned}
\pi_0 &= e \\
\pi_1 &= (1,2) \\
\pi_2 &= (3,4,5) \\
\pi_3 &= (1,2)(3,4,5)
\end{aligned}$$

Note that all the permutations in Γ can be generated using only π_3 (for example $\pi_1 = \pi_3^3$). The orbital partition of Γ is $\{1,\ 2\}$ and $\{3,\ 4,\ 5\}$. If we projected Γ onto the set $\{1,\ 2\}$ we would have the following permutations

$$\begin{aligned}
\pi'_0 &= e \\
\pi'_1 &= (1,2) \\
\pi'_2 &= e \\
\pi'_3 &= (1,2).
\end{aligned}$$

This projection contains all permutations of the set $\{1, 2\}$, so $\texttt{Proj}_{\{1,\ 2\}}(\Gamma) \cong S^2$. Projecting onto the set $\{3,\ 4,\ 5\}$ gives the permutations

$$\begin{aligned}\pi''_0 &= e\\ \pi''_1 &= e\\ \pi''_2 &= (3,4,5)\\ \pi''_3 &= (3,4,5).\end{aligned}$$

This projection does not contain all permutations of the set $\{3,\ 4,\ 5\}$, so $\texttt{Proj}_{\{3,\ 4,\ 5\}}(\Gamma) \ncong S^3$.

2.2 Orbital Branching

The methods discussed in the remainder of this paper apply to 0/1 MILP problems. A subproblem a in the branch-and-bound tree is defined by two sets: the set of variables fixed to zero, F_0^a, the set of variables fixed to one, F_1^a. We let N^a be the set of free variables at node a. We let $\mathcal{F}^a$ denote the feasible set of a and $\mathcal{G}^a$ its symmetry group. As variables are fixed, $\mathcal{G}^a$ changes and needs to be recomputed. For example, suppose x_i and x_j share an orbit at the root node (with π in $\mathcal{G}$ mapping i to j). If variable x_i is fixed to zero and x_j is fixed to one at node a, then for any feasible x at node a, $\pi(x)_i = 0$ and $\pi(x)_j = 1$, meaning that $\pi(x)$ is not feasible in node a, so π is not in $\mathcal{G}^a$.

Orbital branching works as follows. Let $O = \{x_{i_1},\ x_{i_2}, \ldots,\ x_{i_k}\}$ be an orbit of $\mathcal{G}^a$. Rather than branching on the disjunction

$$x_{i_1} = 1 \ \vee \ x_{i_1} = 0, \tag{4}$$

one uses the branching disjunction

$$\sum_{j=1}^{k} x_{i_j} \geq 1 \ \vee \ \sum_{j=1}^{k} x_{i_j} = 0. \tag{5}$$

Note that $\sum_{j=1}^{k} x_{i_j} = 0$ fixes all the variables involved to zero. Symmetry is exploited by the following observation regarding $\sum_{j=1}^{k} x_{i_j} \geq 1$: since all the variables in O are isomorphic, and at least one of them must be equal to one, any variable in O can be arbitrarily chosen to be one. This leads to the symmetry-strengthened disjunction

$$x_{i_1} = 1 \ \vee \ \sum_{j=1}^{k} x_{i_j} = 0. \tag{6}$$

It is easy to see that (6) is a valid disjunction by considering the following. Because the variables are isomorphic, the subproblem generated by fixing x_{i_1} to one will be equivalent to the subproblem generated by fixing x_{i_j} to one for all $j = 2,\ \ldots,\ k$. Therefore, if there is an optimal solution with x_{i_1} equal to

one, then there will be an optimal solution with $x_{i_2} = 1$. If there is an optimal solution that is feasible at a, there will be an optimal solution feasible in one of the subproblems of a.

The strength in orbital branching is that a total of $|O| + 1$ variables are fixed by the branching, instead of the two that traditional branching fixes. Because of this, the larger the orbit, the more likely the branching decision will be strong. Stronger branching decisions will improve the lower bound faster, leading to shorter solution times.

Example 2 This example shows the effectiveness of orbital branching on the Jeroslow problem (1). Figure 1 shows the branch-and-bound tree for the instance where $n = 8$. A pure branch-and-bound approach requires exponentially many (in n) branches, whereas orbital branching requires only a linear number of branches. All nodes except node K are pruned by infeasibility. The LP

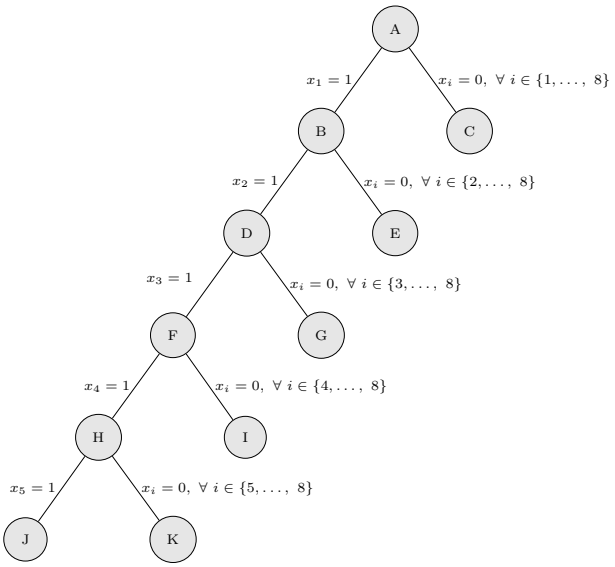


Fig. 1 B&B tree for the Jeroslow problem with orbital branching

solution at node K is integer (and optimal).

3 Modified Orbital Branching

The branching tree in Example 2 is very lopsided. While the unbalanced nature for this example is extreme, it is likely that the branch-and-bound tree will be unbalanced for highly symmetric problems, as the right branch is typically much stronger (fixing $|O|$ variables to zero) than the left branch (fixing one variable to one). This may be problematic because the symmetry group of the left subproblem is typically much smaller than that of the current node.

Thus subsequent branching disjunctions in the left subproblem are likely to be weaker as the smaller symmetry groups lead to smaller orbits. The symmetry group of the right child does not decrease [18], leading to more powerful branches, but this only helps when the right node is not pruned.

In some situations, the branching decision can be modified to create a more balanced branch-and-bound tree. Suppose we branch on orbit $O = \{x_{i_1}, \ldots, x_{i,k}\}$ at subproblem a with symmetry group $\mathcal{G}^a$, and consider the branching disjunction

$$\sum_{j=1}^{k} x_{i_j} \geq b' \ \vee \ \sum_{j=1}^{k} x_{i_j} \leq b' - 1 \tag{7}$$

for any $b' \in \mathbb{Z}^+$. If $\texttt{proj}_O(\mathcal{G}^a)$ is equivalent to $S^{|O|}$, then the above disjunction can be strengthened in a similar way as (5). If an orbit contains at least b' many variables that take the value of 1 and $\texttt{proj}_O(\mathcal{G}^a) \cong S^{|O|}$, then b' variables can arbitrarily be chosen to take the value of 1. Similarly, if at most $b'-1$ variables take the value of 1, then at most $|O| - b' - 1$ variables take the value of 0. If $\texttt{proj}_O(\mathcal{G}^a) \cong S^{|O|}$, these variables can be chosen arbitrarily. Thus disjunction (7) can be strengthened to

$$x_{i_j} = 1 \ \forall \ j \in \{1, \ldots, \ b'\} \ \vee \ x_{i_j} = 0 \ \forall \ j \in \{b', \ b'+1, \ldots, \ |O|\}. \tag{8}$$

While different values of b' can be chosen for (7), the choice $b' = \lceil \sum_{j \in O} x^*_j \rceil$ is natural, where x^* is the solution to the LP relaxation of subproblem a.

Theorem 1 *Let $a = (F^a_0, F^a_1)$ be a node in the branch and bound tree with feasible region $\mathcal{F}^a$. Suppose orbit O with $\texttt{proj}_O(\mathcal{G}^a) \cong S^{|O|}$ was chosen for branching. Child l is formed by fixing $x_{i_j} = 1 \ \forall \ j \in \{1, \ldots, \ b'\}$ and child r is formed by fixing $x_{i_j} = 0 \ \forall \ j \in \{b', \ b'+1, \ldots, \ |O|\}$. For any optimal x^* in $\mathcal{F}^a$, there exists a $\pi \in \mathcal{G}^a$ with $\pi(x^*)$ contained in either $\mathcal{F}^l$ or $\mathcal{F}^r$.*

Proof Assume that $\sum_{j \in O} x^*_j \geq b'$. We construct a $\pi \in \mathcal{G}^a$ such that $\pi(x^*) \in \mathcal{F}^l$.

Let $I = \{j \in O | x^*_j = 1\}$, $\mathcal{I} = \{i \in I \mid b' < i\}$, and $\mathcal{J} = \{j \in \{0, \ldots, \ b'\} \mid j \notin I\}$. By our assumption, we have $|\mathcal{I}| \geq |\mathcal{J}|$.

Let π initially be the identity permutation. For every $j \in \mathcal{J}$, choose a unique element $i \in \mathcal{I}$. Because $\texttt{proj}_O(\mathcal{G}^a) \cong S^{|O|}$, there exists a $\pi_{j,i}$ in $\mathcal{G}^a$ with $\pi_{j,i}(e_j) = e_i$, $\pi_{j,i}(e_i) = e_j$, and $\pi_{j,i}(e_k) = e_k$ for all $k \in O$ not equal to i or j. Amend π such that $\pi \stackrel{\text{def}}{=} \pi \circ \pi_{j,i}$.

By the definition of a group, the resulting π will be an element of $\mathcal{G}^a$ (as it is a composition of elements of $\mathcal{G}^a$). As such, $\pi(x^*)$ is in $\mathcal{F}^a$. Moreover, π maps variables that take the value of "1" in x^* to elements in $\mathcal{J}$ while leaving the variables in $I \setminus \mathcal{I}$ unchanged. As a result, $\pi(x^*)$ satisfies the constraint $\sum_{j \in O} x_j \geq b'$, and is in $\mathcal{F}^l$.

The case of $\pi(x^*)$ in $\mathcal{F}^r$ is proved similarly. $\square$

Example 3 Let us apply modified orbital branching to the Jeroslow problem (1). First, note that the orbit $O_1 = \{1, \ldots, 8\}$ is such that $\texttt{proj}_{O_1}(\mathcal{G}) \cong S^8$, as any permutation of the first 8 variables is a symmetry of the problem. While there are many basic optimal solutions to the LP relaxation, it is easy to verify that all optimal solutions x^* to the LP at the root node have $\sum_{j=1}^{8} x_j^* = 4.5$, hence we use $b' = 5$ and branch on O_1.

The resulting branch-and-bound tree is shown in Figure 2. While node B is infeasible, the LP relaxation at C has $x_1 = x_2 = x_3 = x_4 = x_9 = 1$, an integer (and optimal) solution. Note that for this particular problem, using modified orbital branching is equivalent to adding the symmetry breaking constraints $x_1 \geq x_2 \geq \ldots \geq x_8$.

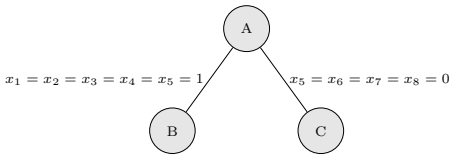


Fig. 2 B&B tree for the Jeroslow problem with modified orbital branching

4 Orbital Branching and Orbitopes

Consider the set of all feasible $m \times n$ 0/1 matrices, where the symmetry acting on the variables consists of all permutations of the columns. A classic example of a problem with this structure is graph coloring. One type of symmetry found in graph coloring arises from permuting colors. If entry i, j equals 1 if and only if vertex i is assigned color j then permuting colors corresponds to permuting two columns of the solution matrix. A common technique used to remove equivalent solutions to the graph coloring problem (as well as other problems) is to restrict the feasible region to matrices with lexicographically non-increasing columns. This idea can be generalized to other problems with this structure.

The convex hull of all $m \times n$ 0/1 matrices with lexicographically non-increasing columns is called a *full orbitope*. A *packing orbitope* is the convex hull of all matrices with lexicographically non-increasing columns with *at most* one 1 per row, and the *partitioning orbitope* is the set with *exactly* one 1 per row. Complete linear descriptions of packing and partitioning orbitopes are given in [12]. While these descriptions require exponentially many inequalities, a polynomial time algorithm can be used to test if a solution of the LP relaxations violates any of the constraints [11]. A complete description of full orbitopes is not known, though an extended formulation is given in [10].

In this section, we show how modified orbital branching can be used to restrict the branch-and-bound search to solutions in the full orbitope. Moreover, we show that modified orbital branching can be used to generate all elements of a full orbitope in polynomial time for a fixed m.

Let $\mathcal{P}(m,n) = \{x \in \mathbb{R}^{m\times n} | x_{i,j} \in \{0,1\}\ \forall i \in \{1,\ldots,m\} \text{ and } j \in \{1,\ldots,n\}\}$. Let $\mathcal{P}_O(m,n) \subset \mathcal{P}(m,n)$ denote the set of matrices with lexicographically non-increasing columns. Suppose the MILP has of the form

$$\min\{\sum_i \sum_j c_{ij}x_{i,j} |\ \sum_i \sum_j a_{ij}^k x_{i,j} \geq b^k\ \ \forall k \in \{1,\ldots,l\},\ \text{and } x \in \mathcal{P}(m,n)\}, \tag{9}$$

and that the symmetry group contains all column permutations of the x variables. As a result of the symmetry, we can restrict our search to be over $\mathcal{P}_O(m,n)$.

While variable orbits can be computed extremely fast if the symmetry group is known, there are no known polynomial-time algorithms to compute the symmetry group of a subproblem in the branch-and-bound tree for a general integer programming problem. However, because of the special structure of orbitopes, computing column symmetries (and thus orbits) can be done in linear time with respect to the number of variables. Permutations that permute columns j and j' are in the symmetry group of the subproblem $a = (F_0^a, F_1^a)$ iff either $x_{i,j}$ and $x_{i,j'}$ are fixed to the same value or both are free for all i in $\{1,\ldots,m\}$.

Even though orbital branching removes a significant proportion of isomorphic solutions from the feasible region, it is still possible for some symmetry to remain unexploited. The extent to which symmetry remains depends on the branching decisions made. With an appropriate branching rule, however, it is possible to remove all column symmetry. One such rule is the *minimum row-index* (MI) branching rule. For this branching rule, $x_{i,j}$ is an eligible branching candidate iff variables $x_{i',j'}$ have already been fixed for all $i' < i$ and all $j' \in \{1,\ldots,n\}$. In other words, variables in row i are only eligible for branching if all variables in rows less than i have already been fixed. The MI branching rule allows orbits to be chosen for branching only if they contain variables eligible for branching. While this may seem highly restrictive, it can be used to ensure that only non-isomorphic solutions are explored. In later sections, we will exploit the rule in order to allow greater flexibility.

Theorem 2 *Suppose the MI branching rule was used at every node in the branch-and-bound tree to generate all non-isomorphic solutions (corresponding to leaf nodes of depth nm in the full branch-and-bound tree). Let $\mathcal{S}$ be the set of solutions generated. We have that $\mathcal{S} = \mathcal{F} \cap \mathcal{P}_O(m,n)$.*

Proof By Theorem 1 we know that $\mathcal{S}$ contains at least one representative from each set of isomorphic solutions, so $|\mathcal{S}| \geq |\mathcal{F} \cap \mathcal{P}_O(m,n)|$. We need only to show that $\mathcal{S} \subseteq \mathcal{F} \cap \mathcal{P}_O(m,n)$ in order to prove this theorem.

Suppose $\mathcal{S} \not\subseteq \mathcal{F} \cap \mathcal{P}_O(m,n)$. Let x be such that x is in $\mathcal{S}$ but not in $\mathcal{F} \cap \mathcal{P}_O(m,n)$. Clearly, we have that $x \in \mathcal{F}$. As x is not in $\mathcal{P}_O(m,n)$, it must contain two adjacent columns, j and $j+1$, with column $j+1$ lexicographically larger than column j. Let i be the first row where columns j and $j+1$ differ. We have $x_{i,j} = 0$ and $x_{i,j+1} = 1$.

By definition of $\mathcal{S}$, there is a node of depth mn in the branch-and-bound tree representing solution x. Let subproblem a be the earliest ancestor of this subproblem where either $x_{i,j}$ or $x_{i,j+1}$ was fixed by branching. At node a, the fixed variables in column j are identical to those in column $j+1$. If $x_{i,j}$ was fixed to zero by branching, then $x_{i,j+1}$ must be in the orbit branched upon. In that case, the branching disjunction that fixed $x_{i,j}$ to zero would have also fixed $x_{i,j+1}$ to zero (since $j < j+1$). Similarly, if $x_{i,j+1}$ was fixed to one by branching, then $x_{i,j}$ must have been fixed to one. This branching disjunction would have removed x from both children's feasible region, so no such x can exist. □

Theorem 2 can be extended to MILPs where only a subset of the variables can be expressed as 0/1 matrices.

4.1 The Complexity of Enumerating the Elements of an Orbitope

In this section we show that for fixed m, there are polynomially many solutions in $\mathcal{P}_O(m,n)$. As a consequence of this, modified orbital branching can enumerate all feasible solutions in polynomial time (for a fixed m).

Theorem 3 *For fixed m, the number of solutions in $\mathcal{P}_O(m,n)$ grows polynomially in n.*

Proof We use a combinatorial argument. Let $\mathcal{B}$ be the (numbered) collection of 0/1 m-vectors. We represent each feasible solution in $\mathcal{P}_O(m,n)$ as a collection of n of these m-vectors (with duplication). Note that $|\mathcal{B}| = 2^m$. Using a "stars-and-bars" argument, we know that there are $\binom{n+|\mathcal{B}|-1}{|\mathcal{B}|}$ ways to choose n elements from a set of $\mathcal{B}$ elements with replacements. Since $\binom{n+|\mathcal{B}|-1}{|\mathcal{B}|} = O(n^{|\mathcal{B}|})$ for large n, the result follows. □

Next, we need to show that the branch-and-bound tree grows polynomially as n increases. This is easy to see by observing that at any subproblem in the branch-and-bound tree, a feasible solution to $\mathcal{P}_O(m,n)$ can be found by fixing all free variables to zero. As a result, there can be no more than $\binom{n+|\mathcal{B}|}{|\mathcal{B}|}$ many nodes at any depth in the tree. The depth of the branch-and-bound tree is bounded above by $mn+1$, so there can be at most $mn\binom{n+|\mathcal{B}|}{|\mathcal{B}|}$ many nodes. At each subproblem, only the orbits and the minimum row-index need to be computed. These can also be done in polynomial time.

4.2 Strength of LP Relaxation

Theorem 2 shows that modified orbital branching *eventually* removes isomorphic solutions from the feasible solutions. Nevertheless one concern with using modified orbital branching is that the LP relaxations might be weaker than

if the isomorphic solutions were removed at the root node and weaker relaxations might lead to larger branch-and-bound trees. This might be the case in bin packing problems, where the convex hull of the bin-packing polytope is known [11]. Usually, though, symmetry-breaking constraints don't exploit integrality, and as a result, tend to be very weak. For instance, in his paper [7], Friedman showed that all isomorphic solutions can be removed from the LP relaxation's feasible region by adding the constraints

$$\sum_{k=1}^{m} 2^{m+1-k} x_{k,j} \geq \sum_{k=1}^{m} 2^{m+1-k} x_{k,j+1} \ \forall j \in \{1, \ldots, \ n-1\}. \tag{10}$$

These constraints force the columns of x to be lexicographically non-increasing. It is clearly not practical to add these constraints to any problem of reasonable size, as the 2^{m+1-k} term may cause numerical instability. However, even if it were, these constraints would not improve the LP relaxation at the root node.

Another option for symmetry-breaking inequalities is to add *column inequalities* [12]. Like the Friedman inequalities, column inequalities also enforce lexicographically decreasing columns by adding the (more computationally stable) constraints

$$x_{i,j} \leq \sum_{k=1}^{i} x_{k,j-1} \ \forall i \in 1, \ \ldots, m, \ \forall j \in 2, \ \ldots, \ n. \tag{11}$$

Theorem 4 *For any subproblem of* (9) *formed by using modified orbital branching with a MI branching rule, the solution to the LP relaxation does not change when either constraints* (10) *or* (11) *are added to the formulation.*

Proof A well known result from [8] states that for a linear program with symmetry group $\mathcal{G}$, there exists an optimal solution with $x_i = x_{\pi(i)}$ for all $\pi \in \mathcal{G}$. Because all column permutations are symmetries, there will always be optimal LP solution at the root node with $x_{i,j} = x_{i,j+1}$ for all appropriate i and j. This solution is not removed by the inequalities described in (10) or by those described in (11). □

Note however that constraints (10) and (11) may be strengthened by considering the integrality of the variables in order to generate tighter cuts. Theorem 4 does not apply to the resulting integrality-based constraints.

4.3 Relaxing the Branching Rule

Theorem 2 requires the MI branching strategy. Example 4 illustrates how isomorphic solutions can remain if the MI branching rule is not used.

Example 4 Figure 3 shows a branching tree for a problem where x is a 2×2 0/1 matrix whose symmetry contains the permutation of the columns of x.

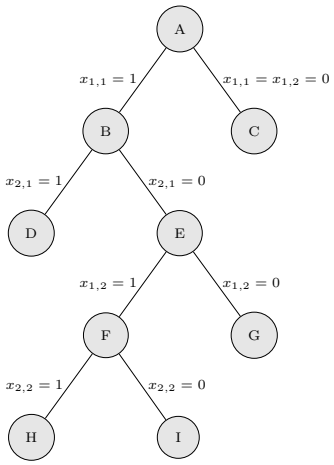


Fig. 3 Symmetry remaining after modified orbital branching

Note that the MI branching rule is not used at every node in the tree. The solution represented by subproblem H is

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \tag{12}$$

which does not have lexicographically non-increasing columns. This solution is equivalent to

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \tag{13}$$

which is feasible in node D.

Why does modified orbital branching (without MI branching) fail to prune the solution at node H from the feasible region? The problem starts at node B. There is no symmetry in node B, as the first column of x contains one fixed variable and the second column does not. However, symmetry is present in a subset of $\mathcal{F}_B$, namely $\{x \in \mathcal{F}_B | x_{1,2} = 1\}$. Fixing $x_{1,2}$ to one at node B (as is done with the MI branching rule) would reincorporate the column symmetry into the subproblem. However this is not done in Figure 3. Further branching also does not exploit this symmetry, and as a result, the search explores more solutions than is necessary.

Figure 4 shows the branch-and-bound tree when the MI branching rule is used. Note that the column symmetry is exploited in subproblem D', leading to a nontrivial orbital branch. As a result of this branch, solution (12) is removed from the search space.

Making a rigid application of a branching rule such as MI branching may hinder the performance of the solver because different strategies to choose branching variables can have a large impact on the time required to solve the problem. Fortunately, a few tricks can be implemented to add more flexibility in branching. Similar to the strategy used in [15], row indices can be permuted

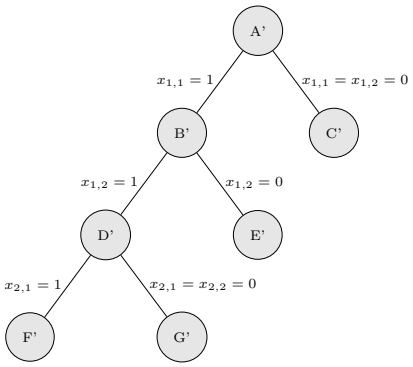


Fig. 4 No symmetry remaining with the MI rule

to favor branching. For example, suppose the current subproblem contains solutions of the type

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \\ x_{3,1} & x_{3,2} \\ x_{4,1} & x_{4,2} \end{pmatrix}. \tag{14}$$

The MI branching rule requires that either $x_{3,1}$ or $x_{3,2}$ be chosen for branching. However, by reindexing the rows, we can create a new subproblem with solutions of the type

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \\ x'_{3,1} = x_{4,1} & x'_{3,2} = x_{4,2} \\ x'_{4,1} = x_{3,1} & x'_{4,2} = x_{3,2} \end{pmatrix} \tag{15}$$

and now either $x'_{3,1}$ or $x'_{3,2}$ (representing $x_{4,1}$ and $x_{4,2}$) can be chosen for branching. Note that a row cannot be reindexed after any variable in that row has been fixed by branching. Similar to isomorphism pruning [16], this leads to a reordering of the rows for each subproblem in the branch-and-bound tree. This reordering can be kept track of by using a *rank vector*. The rank vector $\mathbf{r}^a \in \mathbb{R}^m$ for subproblem a denotes the order in which rows were branched on. $\mathbf{r}^a_i = r < m+1$ designates that row i was the rth row branched on, whereas $\mathbf{r}^a_i = m+1$ indicates that no variable in row i has been fixed by branching. The *minimum row-rank* (MRR) branching rule states that if there is a row with rank less then $m+1$ that contains a free variable, a variable from that row must be chosen for branching (only one such row will exist). The resulting solutions using the MRR branching rule may not have lexicographical solutions, but the space explored will not contain any isomorphic solutions. A corollary to Theorem 2 is:

Corollary 1 *Suppose the MRR branching rule was used at every node in the branch-and-bound tree to generate all non-isomorphic solutions (corresponding to leaf nodes of depth nm in the full branch-and-bound tree). Let $\mathcal{S}$ be the set of solutions generated. We have that $|\mathcal{S}| = |\mathcal{F} \cap \mathcal{P}_O(m,n)|$.*

Even more flexibility in branching can be obtained by recognizing when symmetry will never be reincorporated. This is the case in the partial solution shown in (15). Independent of the unknown variables, column two will always be lexicographically smaller than column one. This is easily seen by recognizing that the first difference between the solutions in the columns are at row two, where column one has a "1" and column two has a "0". In this case, variables can be branched on in any order and still guarantee that only nonisomorphic solutions are explored. We need to test if and when this is the case.

Suppose we wish to branch on orbit $O_{i,j}$ containing the k elements $O_{i,j} = \{x_{i,j},\ x_{i,j+1}, \ldots,\ x_{i,j+k-1}\}$ at subproblem a. Let $\mathbf{r}^a$ be the rank vector at a. We say that $O_{i,j}$ is *left closed with respect to* $\mathbf{r}^a$ if $j = 1$ or column j (reindexed by $\mathbf{r}^a$) is guaranteed to be lexicographically smaller than column $j-1$. Left closure is easily determined by finding the first (after reindexing) row i' with $x_{i',j-1}$ fixed to a different value than $x_{i',j}$. If $x_{i',j-1} \in F_1^a$ and $x_{i',j} \in \mathcal{F}_0^a$, then $O_{i,j}$ is left closed. Similarly, $O_{i,j}$ is *right closed with respect to* $\mathbf{r}^a$ if either $j = n$ or $O_{i,j+k+1}$ is left closed (the jth column of the matrix will always be lexicographically larger than the $j+k+1$st column). Orbit $O_{i,j}$ is said to be *closed* if it is both left closed and right closed. We define the *closure* of $O_{i,j}$, $O_{i,j}^C$, to be the minimal set $\{x_{i,j'}, x_{i,j'+1}, \ldots, x_{i,j''}\}$ such $j' \leq j$, $j'' \geq j+k-1$, the orbit containing $x_{i,j'}$ is left closed, and the orbit containing $x_{i,j''}$ is right closed. Note that closure depends only on the columns, not the rows. If $O_{i,j}$ is right/left closed, then orbit $O_{i+1,j}$ will also be right/left closed.

The notion of closure is meant to indicate whether additional variable fixings can be used to increase the size of a given orbit. For instance, if $O_{i,j} = \{x_{i,j},\ x_{i,j+1}, \ldots,\ x_{i,j+k-1}\}$ is not left closed, then fixing some free variables might create a new symmetry group where $\{x_{i,j-1}, x_{i,j},\ x_{i,j+1}, \ldots,\ x_{i,j+k-1}\}$ is an orbit. Similarly, if $O_{i,j}$ is not right closed, then variable fixings might cause the orbit to become larger. Recall from Example 4 that when symmetry enters the problem as a result of fixings, then some isomorphic solutions may remain (necessitating the use of the MRR branching rule). However, if $O_{i,j}$ is left closed, then independent of additional fixings, column $j-1$ will be lexicographically larger than any column in $\{j, j+1, \ldots, j+k-1\}$. Similarly, if $O_{i,j}$ is right closed, then column $j+k$ will be lexicographically smaller than any column in $\{j, j+1, \ldots, j+k-1\}$. The notion of closure can be useful in identifying when the MI (or MRR index) branching rule can be ignored. With that in mind, we create the *relaxed minimum-index* (RMI) branching rule. An orbit $O_{i,j}$ with closure $\{x_{i,j'}, x_{i,j'+1}, \ldots, x_{i,j''}\}$ is eligible for branching if and only if $x_{u,v}$ is fixed for all $u < i$ and all $j' \leq v \leq j''$.

Theorem 5 *Suppose the RMI branching rule was used at every node in the branch-and-bound tree to generate all non-isomorphic solutions. Let $\mathcal{S}$ be the set of solutions generated. We have that $\mathcal{S} = \mathcal{F} \cap \mathcal{P}_O(m,n)$.*

Proof This proof closely follows the proof of Theorem 2. In the proof to Theorem 2, we argue that if $\mathcal{S} \neq \mathcal{F} \cap \mathcal{P}_O(m,n)$, then there must exist an $x \in \mathcal{S}$ that contains columns j and $j+1$ where column $j+1$ is lexicographically larger than column j. Let i be the minimum row index where $x_{i,j} < x_{i,j+1}$, i.e., $x_{i,j} = 0$

and $x_{i,j+1} = 1$. Let a be the earliest ancestor of the nodes representing the solution x where either $x_{i,j}$ or $x_{i,j+1}$ was fixed by branching.

The proof for Theorem 2 relied on the fact that the orbit at a containing $x_{i,j}$ also contains $x_{i,j+1}$. We must prove that with the RMI branching rule variables $x_{i',j}$ and $x_{i',j+1}$ are fixed at a for all $i' < i$. Because of our choice of a, there is no $i' < i$ with $x_{i',j} > x_{i',j+1}$, so the closure of the orbit containing $x_{i,j}$ must also contain $x_{i,j+1}$. As a result, if $x_{i,j}$ and $x_{i,j+1}$ are eligible for branching, then $x_{i',j}$ and $x_{i',j+1}$ must be fixed for all $i' < i$, implying that $x_{i,j}$ and $x_{i,j+1}$ are in the same orbit at node a. The branching disjunction must either set $x_{i,j}$ to 1 or $x_{i,j+1}$ to 0, contradicting our choice of i. □

Example 5 Consider the subproblem with the following fixed variables

$$\left(\begin{array}{c|c|c|c|c|c} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & x_{3,4} & 0 & 0 \\ 0 & x_{4,2} & x_{4,3} & x_{4,4} & x_{4,5} & 0 \\ 1 & x_{5,2} & x_{5,3} & x_{5,4} & x_{5,5} & x_{5,6} \\ x_{6,1} & x_{6,2} & x_{6,3} & x_{6,4} & x_{6,5} & x_{6,6} \\ x_{7,1} & x_{7,2} & x_{7,3} & x_{7,4} & x_{7,5} & x_{7,6} \end{array}\right). \tag{16}$$

The symmetries in this subproblem permute columns 2 and 3.

Let $O_{6,1}$ represent the variable orbit $\{x_{6,1}\}$. $O_{6,1}$ is trivially left closed (because it contains a variable from the leftmost column). It is also right closed because column 2 can never be lexicographically larger than column 1 (since $x_{2,1} = 1$ and $x_{2,2} = 0$). Since orbit $O_{6,1}$ is closed with $x_{i',1}$ fixed for all $i' < 5$, it is eligible for branching using the RMI branching rule.

$O_{4,2} = \{x_{4,2}, x_{4,3}\}$ is left closed (for the same reasons $O_{6,1}$ is right closed), but not right closed. Notice that if $x_{3,4}$ were fixed to 1, then $\{x_{4,2}, x_{4,3}, x_{4,4}\}$ would be a larger orbit containing $O_{4,2}$.

$O_{3,4} = \{x_{3,4}\}$ is neither left closed nor right closed, as fixing $x_{3,4}$ to 1 creates a larger orbit (by adding variable found to the left of $x_{3,4}$) containing $\{x_{3,2}, x_{3,3}, x_{3,4}\}$, and it is not right closed as fixing $x_{3,4}$ to 0 creates the larger orbit $\{x_{3,4}, x_{3,5}\}$ (note that the larger orbits can contain fixed variables).

$O_{4,5}$ is not left closed (by the reasons mentioned above), but it is right closed, as $x_{1,5} = 1$ and $x_{1,6} = 0$.

The closure of $O_{4,2}$, and also the closure of $O_{4,4}$ and $O_{4,5}$, is equal to $\{x_{4,2}, x_{4,3}, \ldots, x_{4,5}\}$, which is not eligible for branching, as $x_{3,4}$ is a free variable. In fact, the orbit containing $x_{3,4}$ is the only orbit in columns 2 through 5 that is eligible for branching using the RMI rule.

The orbit $O_{5,6} = \{x_{5,6}\}$ is both right closed and left closed. Since $x_{i',6}$ is fixed for all $i' < 5$, it is eligible for branching under the RMI branching rule.

Note that the MI branching rule would require that the orbit containing $x_{3,4}$ be chosen for branching. The RMI branching rule adds more flexibility by allowing the solver to branch on the orbit containing $x_{6,1}$, $x_{3,4}$, or $x_{5,6}$.

Similar to the MI branching rule, the RMI rule can be relaxed by allowing rows to be reordered. We define the *relaxed minimum-rank index* (RMRI)

branching rule as follows. Let $R^a \in \mathbb{R}^{m \times n}$ be a matrix whose ith column corresponds to a rank vector associated with column j of x at node a in the branch-and-bound tree, where $R^a_{i,j} = k \leq m$ implies that, amongst all the variables in column j, variable $x_{i,j}$ was the kth variable fixed by branching. If $x_{i,j}$ is a free variable at node a, $R^a_{i,j}$ takes the value $m+1$. The *minimum rank* of a set of variables O at node a, R^a_O is equal to the rank of its smallest element, i.e. $R^a_O = \{\min R^a_{i,j} \mid x_{i,j} \in O\}$. An orbit of free variables is branchable under the RMRI branching rule if and only if the rank of its closure is smaller than any other orbit closure spanning the same set of columns. A consequence of this rule is that if an orbit closure contains both free variables and variables fixed by branching, then an orbit contained in that closure must be chosen for branching. Corollary 2 is a similar extension to Theorem 5 as Corollary 1 is to Theorem 2.

Corollary 2 *Suppose the RMRI branching rule was used at every node in the branch-and-bound tree to generate all non-isomorphic solutions. Let $\mathcal{S}$ be the set of solutions generated. We have that $|\mathcal{S}| = |\mathcal{F} \cap \mathcal{P}_O(m,n)|$.*

Example 6 Consider again (16) in the previous example. Both sets $\{x_{6,1}\}$ and $\{x_{7,1}\}$ are closed orbits, and contain variables with rank of 8 (with respect to R^a). As such, both orbits are eligible for branching.

Consider any orbit containing a variable in any of columns 2 through 5. Suppose orbit $O_{i,j}$ contains $x_{i,j}$, where j is between 2 and 5. We have that $O^C_{i,2} = \{x_{i,2}, \ldots, x_{i,5}\}$. As columns 2 through 5 have no variables fixed in rows 4 through 7, $R^a_{i,j} = 8$ for all $j \in \{2,\ldots,5\}$, and $i \in \{4,\ldots,7\}$. As a result, $R_{O^C_{4,2}} = R_{O^C_{5,2}} = R_{O^C_{6,2}} = R_{O^C_{7,2}} = 8$. However, $O^C_{3,2}$ contains elements with rank 3, ($x_{3,2}$, $x_{3,3}$, and $x_{3,5}$), and one with rank 4, ($x_{3,4}$), so $R^O_{3,2} = 3$. Even though $O^C_{1,2}$ and $O^C_{2,2}$ have ranks smaller than $O^C_{3,2}$, they do not contain any free variable orbits. Since $\{x_{3,3}\}$ is the only free variable orbit in $O^C_{3,2}$, it is eligible for branching.

The orbits $\{x_{5,6}\}$, $\{x_{6,6}\}$, and $\{x_{7,6}\}$ are all closed and all have a rank of 8 with respect to R^a. As a result, all of these orbits are eligible for branching under the RMRI branching rule.

Algorithm 1 summarizes how to choose the appropriate branching orbit for orbitopes.

Recall that for packing (partitioning) problems, solutions can be represented as 0/1 matrices with at most (exactly) one 1 per column, such that all permutations of the column indices are symmetries. It is easy to see that in this case every orbit in every subproblem will be closed, meaning that isomorphic solutions can be removed from the search without requiring any branching restrictions.

4.4 Implementation

Algorithm 1 describes how to use modified orbital branching to ensure that the branching process explores only non-isomorphic solutions. One implementation

Algorithm 1 Selecting Branching Orbit at Subproblem $a = (F_0^a, F_1^a)$

INPUT: F_1^a, F_0^a, rank matrix R^a for all j, and candidate branching orbit $O_{i,j} = \{x_{i,j}, \ldots, x_{i,j+k}\}$.
OUTPUT: Branchable orbit
If $O_{i,j}$ **is left closed**
  If $O_{i,j}$ **is right closed**
    Branch on $O_{i,j}$
  Else
    Let r be the smallest ranked index where only one of $x_{r-1,j}$ and $x_{r-1,j+k+1}$ is in N^a.
    If $x_{r,j} \in F_1^a$
      Branch on $O_{r,j+k+1}$
    If $x_{r,j+k+1} \in F_0^a$
      Branch on $O_{r,j}$
Else
  Let r be the smallest ranked index where only one of $x_{r-1,j-1}$ and $x_{r-1,j}$ is in N^a
    If $x_{r,j-1} \in F_1^a$
      Branch on $O_{r,j}$
    If $x_{r,j} \in F_0^a$
      Branch on $O_{r,j-1}$

difficulty, however, is the use of the rank vector. The rank vector describes the branching decisions made on the path from the root node to the current problem in the branch-and-bound tree. Some MILP solvers do not keep track of what the actual tree looks like, and storing the rank vector for each subproblem may consume a lot of memory. Fortunately, the rank vector is not necessary.

The rank vector is used to determine two things. First, it is needed to determine if a given orbit is closed and branchable. Second, it is used to find a branchable orbit if the initial orbit is not. Suppose that all variable fixings occur only through RMRI branching (no reduced cost fixing or other logical implications). Using Theorem 6, it is easy to determine if a given orbit $O_{i,j} = \{x_{i,j}, \ldots x_{i,j+k}\}$ is closed (both left and right) without knowing the rank vector.

Theorem 6 *Let $O_{i,j} = \{x_{i,j}, \ldots x_{i,j+k}\}$ be an orbit in $\mathcal{G}^a$ with $1 < i$ and $j + k < n$. $O_{i,j}$ is left closed with respect to R^a if and only if there is a row i' such that $x_{i',j-1} \in F_1^a$ and $x_{i',j} \in F_0^a$. Similarly, $O_{i,j}$ is right closed with respect to R^a if and only if there is a row i' such that $x_{i',j} \in F_1^a$ and $x_{i',j+k+1} \in F_0^a$.*

Proof Let i' be the smallest ranked row with $x_{i',j-1} \in F_1^a$ and $x_{i',j} \in F_0^a$. Aiming at a contradiction, assume that $O_{i,j}$ is not left closed. Because $O_{i,j}$ is not left closed, there must be a row r with rank smaller than i' with either

1. $x_{r,j-1}, x_{r,j} \in N^a$,
2. $x_{r,j-1} \in \mathcal{F}_1^a$ and $x_{r,j} \in N^a$,
3. $x_{r,j-1} \in N^a$ and $x_{r,j} \in F_0^a$.

In any of these cases, there is a free variable in a column with rank less then i', meaning that neither $x_{i',j-1}$ nor $x_{i',j}$ could have been fixed by branching in Algorithm 1 using the RMRI branching rule.

A similar argument can be made to show that if there exists a row i' with $x_{i',j} \in F_1^a$ and $x_{i',j+k+1} \in F_0^a$, then $O_{i,j}$ must be right closed. □

Suppose that the orbit $O_{i,j}$ is not closed. We must then test if $O_{i,j}$ is branchable, and if it is not, find a branchable orbit. This is done using Theorem 7.

Theorem 7 *Let a be a node in a branch-and-bound tree generated by using the RMRI branching rule at every node. Consider orbit $O_{i,j}$, with closure $O^C_{i,j} = \{x_{i,j'}, \ldots, x_{i,j''}\}$. If $O_{i,j} \neq O^C_{i,j}$, then there is exactly one i' such that the set $O^C_{i',j} = \{x_{i',j'}, \ldots, x_{i',j''}\}$ contains both a fixed variable (by branching) and a free variable. Moreover, any orbit $O_{k,l} \in O^C_{i',j}$ containing free variables is branchable.*

Proof Suppose no such i' exists. WLOG we can assume that $O_{i,j}$ is not left closed (i.e., $j' < j$). Because $O_{i,j}$ is not left closed, there is no i' with $x_{i',j'} = 1$ and $x_{i',j} = 0$. If there is an i' with $x_{i',j'} = 0$ and $x_{i',j} = 1$, then column j will be lexicographically larger (with respect to the rank vectors) than column j', which cannot happen using the RMRI branching rule. As a result, for any i', either $x_{i',j'}$ and $x_{i',j}$ are fixed to the same value or they are both free variables (note that the nonexistance of i' dissallows one variable to be fixed and the other variable to be free). In this case, $x_{i,j'}$ shares an orbit with $x_{i,j}$, contradicting the fact that $j' < j$.

Suppose instead that we have i' and i'', with $i' \neq i''$, where $O^C_{i',j} = \{x_{i',j'}, \ldots, x_{i',j''}\}$ and $O_{i'',j} = \{x^C_{i'',j}, \ldots, x_{i'',j''}\}$ contains both a fixed variable and a free variable. Assume $R^a_{O^C_{i',j}} < R^a_{O^C_{i'',j}}$, i.e., a variable in $O^C_{i',j}$ was branched on before any variables in $O^C_{i'',j}$. Let b be the node where the first variable in $O^C_{i'',j}$ was fixed by branching. We know that $R^b_{O^C_{i'',j}} = m + 1$, but $O_{i',j}$ contains a free variable and $R^b_{O^C_{i',j}} < m+1$. Thus, branching on any orbit in $O_{i'',j}$ would violate the RMI branching rule. □

Theorem 6 relies on the fact that variables are only fixed by branching. In reality, a solver will attempt to fix variables by a variety of methods, such as reduced-cost fixing or strong-branch fixing. These fixings make it difficult to determine the appropriate branching orbit. For example, two consecutive columns may have many rows where they take different values. Fortunately, though, the structure of the modified branching algorithm can be studied in order to return good branchable orbits. Recall from the algorithm that if $O_{i,j}$ is not left closed, then the row index is chosen by looking for a row with either $x_{i,j-1} \in F_1^a$ and $x_{i,j} \in N^a$ or $x_{i,j-1} \in N^a$ and $x_{i,j} \in F_0^a$. If no such row exists, then additional variables can be fixed in a way that makes $O_{i,j}$ a larger orbit. If there are two or more such rows, then any of these rows can be arbitrarily chosen without affecting the correctness of the branching disjunction, as the branch is then equivalent to a modified orbital branching without a branching rule. However, this may lead to isomorphic subproblems and solutions.

5 Orbital Branching and the Unit Commitment Problem

We demonstrate the effectiveness of modified orbital branching by applying it to the unit commitment (UC) problem. The UC problem is a fundamental problem in the operation of power systems. The purpose of UC is to minimize the total cost of power generation by finding an optimal power production schedule for each generator while ensuring that demand is met and that the system operates safely and reliably. In the context of practical system operation, the instances of UC that must be solved are often challenging because they are large-scale and require significant computational time to solve, while the time available to solve a UC model is a hard limitation. The survey paper [1] provides an introduction to UC from the point of view of optimization and a summary of the recent developments in the literature.

The solution of UC problems can be particularly difficult when there are many identical generators, resulting in a problem with many symmetries. In practice, the MILP formulations were at first solved using Lagrangian relaxation. However, even without branch-and-bound, symmetry still causes numerical difficulties in Lagrangian relaxation methods. The paper [21] addressed this issue by developing a surrogate subgradient method to update second level prices. At present, most system operators use MILP solvers whose performance may benefit from effective symmetry-breaking methods.

One way to remove symmetry from the UC problem's formulation is to aggregate all identical generators into a single generator. This is typically done for combined cycle plants that can have two or three identical combustion turbines. While this might be effective in reducing the number of variables, aggregating generator variables may be very difficult, and some of the physical requirements may be difficult to enforce. If the UC solution only specifies the number of generators operating at each time period as well as the total power produced by those generators, it might be difficult to determine which generators produce how much power at each time period. A comparison between aggregating combined cycle generators and modeling them as individual units can be found in [13].

We now describe the formulation of UC that we use for the computational experiments in this paper. This formulation assumes that there are multiple generators with the same characteristics; specifically generators are divided into K classes such that generators in the same class can be treated as identical.

Given a set of power generators and set of electricity demands, the UC problem minimizes the total production cost of the power generators subject to the constraints that

1. the demand is met, and
2. the generators operate within their physical limits, i.e.,
   (a) the power output level of a generator may not change too rapidly (ramping constraints), and
   (b) when a generator is turned on (off), it must stay on (off) for a minimum amount of time (minimum up / downtime constraints).

We use the three-binaries-per-generator-hour formulation based on [2]. This model contains three sets of binary variables at every time period: those representing the on/off status of each generator at the time period, those representing if each generator is started up in the time period, and those representing if each generator is shut down in the time period.

The problem data are as follows:

- T : Number of time steps (h).
- D_t: Demand at time period t (MW)
- R_t: Generation reserves required at time t (MW)
- K : Set of generator types.
- G^k : Set of generators of type k.
- n_k : Number of generators of type k
- b^k_{low}, b^k_{high} : Coefficients for the piecewise linear approximation of the cost function for a generator of type k ($/MW).
- P^k_{low}, P^k_{high} : Marginal cost coefficients for piecewise linear approximation of the cost function for a generator of type k ($/MW).
- $\underline{P}^k$, $\overline{P}^k$: Minimum and maximum generator limits for generator of type k (MW).
- RD^k : Ramp-down rate of generator of type k (MW/h).
- RU^k : Ramp-up rate of generator of type k (MW/h).
- SD^k : Shutdown limit of generator of type k (MW).
- SU^k : Startup level of generator of type k (MW).
- UT^k : Minimum uptime for generator of type k (h).
- DT^k : Minimum downtime for generator of type k (h).

The variables are:

- $c^k_{t,g}$: Operating cost for generator g of type k at time t ($).
- $p^k_{t,g}$: Power produced at generator g of type k at time t (MW).
- $u^k_{t,g}$: On/off status of generator g of type k at time t.
- $v^k_{t,g}$: Startup status of generator g of type k at time t.
- $w^k_{t,g}$: Shutdown status of generator g of type k at time t.

The UC problem formulation is:

$$\min \quad \sum_{t=1}^{T} \sum_{k \in K} \sum_{g \in G^k} c^k_{t,g} \tag{UC}$$

subject to

$$c^k_{t,g} \geq P^k_{low} p^k_{t,g} + b^k_{low} u^k_{t,g}, \ \forall g \in G^k, \forall k \in K, t = 1, \ldots, T \tag{17}$$

$$c^k_{t,g} \geq P^k_{high} p^k_{t,g} + b^k_{high} u^k_{t,g}, \ \forall g \in G^k, \forall k \in K, t = 1, \ldots, T \tag{18}$$

$$\sum_{k \in K} \sum_{g \in G^k} p^k_{t,g} \geq D_t + R_t, \ t = 1, \ldots, T \tag{19}$$

$$\underline{P}^k u^k_{t,g} \leq p^k_{t,g} \leq \overline{P}^k u^k_{t,g}, \ \forall g \in G^k, \forall k \in K, t = 1, \ldots, T \tag{20}$$

$$p^k_{t,g} - p^k_{t-1,g} \le RU^k u^k_{t-1,g} + SU^k v^k_{t,g},\ \ \forall g \in G^k, \forall k \in K, t = 1, \ldots, T \quad (21)$$

$$p^k_{t-1,g} - p^k_{t,g} \le RD^k u^k_{t,g} + SD^k w^k_{t,g},\ \ \forall g \in G^k, \forall k \in K, t = 1, \ldots, T \quad (22)$$

$$\sum_{\ell=t-UT^k+1,\ \ell\ge 1}^{t} v^k_{\ell,g} \le u^k_{t,g},\ \ \forall g \in G^k, \forall k \in K, t = 1, \ldots, T \quad (23)$$

$$u^k_{t,g} + \sum_{\ell=t-DT^k+1,\ \ell\ge 1}^{t} w^k_{\ell,g} \le 1,\ \ \forall g \in G^k, \forall k \in K, t = 1, \ldots, T \quad (24)$$

$$u^k_{t-1,g} - u^k_{t,g} + v^k_{t,g} - w^k_{t,g} = 0 \forall g \in G^k,\ \ \forall k \in K, t = 1, \ldots, T \quad (25)$$

$$c^k_{t,g}, p^k_{t,g} \in \mathbb{R}^+,\ \ \forall g \in G^k, \forall k \in K, t = 1, \ldots, T \quad (26)$$

$$u^k_{t,g},\ v^k_{t,g},\ w^k_{t,g} \in \{0, 1\},\ \ \forall g \in G^k, \forall k \in K, t = 1, \ldots, T \quad (27)$$

The cost function of a generator is generally assumed to be a quadratic function and it is customary to approximate it with a piecewise linear function as we do here. This approximation is done using the two line segments in constraints (17) and (18) that are derived from two tangent lines of the quadratic cost function (representing a low cost and a high cost) strengthened using the method described in [5,6]. Constraint (19) ensures that enough power is produced to meet demand. (We assume demand is known a priori.) Constraints (20) through (25) ensure that each generator's production schedule is feasible [2]. Constraint (20) ensures that each generator's production is within its normal operating limit. Constraints (21) and (22) are ramping constraints ensuring that the output of each generator does not change too rapidly. Constraints (23) and (24) are minimum up and downtime constraints. For these constraints, a negative time index corresponds to a generator's status before the start of the timing horizon. Similarly, a time index larger than T represents the generator's status after the planning horizon has expired. Constraint (25) is a logical constraint, ensuring that the $v^k_{t,g}$ variable must take the value of 1 if generator g is turned on at time t, and that $w^k_{t,g}$ must take the value of 1 of it is turned off. This form of the minimum on and off time constraints was proposed in [19], and tightened ramping constraints were given in [17].

We use the three-binaries-per-generator-hour formulation based on [2]. This model contains three sets of binary variables at every time period: those representing the on/off status of each generator at the time period, those representing if each generator is started up in the time period, and those representing if each generator is shut down in the time period. Since the startup/shutdown status can be easily determined if the on/off status is known, in our discussion we focus only on those variables indicating if the generator is on or off. This is done for notational convenience. It is common in the power systems literature to relax the integrality constraints of the startup and shutdown variables.

The binary variables in UC can be expressed as a series of 0/1 matrices. For example, we can let U^k to be the $T \times n_k$ 0/1 matrix where $U^k_{t,g} = u^k_{t,g}$. Similarly, one can construct C^k, P^k, V^K, and W^k. Because any two generators

in class k are identical, their production schedules can be permuted to form identical (isomorphic) solutions. Permuting the schedules of two generators of the same type is equivalent to permuting their respective columns in each of U^k, C^k, P^k, V^K, and W^k. As all generators of the same class are identical, any such permutation of columns will be a symmetry, so long as the permutation is applied to all the matrices. For the sake of notational simplicity, we will only consider the variables in U^k, those representing the on/off status of each generator of class k. We assume that any symmetry that permutes columns of U^k will also permute columns of C^k, P^k, V^K, and W^k. This is appropriate for the UC problem because the remaining binary variables can be uniquely determined if the U^k variables are known, and breaking the symmetry of the U^k variables will break all the symmetry between the generators of class k.

The following subsections describe computational tests related to symmetry breaking and the UC problem. All tests were performed on a Dell PowerEdge T620 with two Intel Xeon E5-2670 2.60GHz processors and eight 32 GB RDIMM chips. It is running Ubuntu version 12.10. Instances were solved one at a time with no other (significant) program running concurrently. The MILP problems were solved using CPLEX version 12.5.1.0. All instances were solved to within 0.1% optimality and the number of available threads was set to 1.

To test the impact of modified orbital branching on the UC problem, we generated 25 instances of the UC problem based on generator characteristics described in [3]. The test problems in [3] have 8 unique generators described in Tables 1 and 2. The quadratic cost functions were approximated by a piecewise linear function. The marginal cost of power produced in the bottom half of the possible production quantities is P^k_{low}, while the marginal cost of producing power above the midpoint is P^k_{high}. If a generator has been off for fewer than t^{cold}_j hours, it pays a startup cost of hc_j. Otherwise, its startup cost is cc_j. Generation reserves were set to be 3% of demand.

By replicating generators and scaling the hourly demand appropriately, we generated instances of varying sizes, each primarily containing generators of type 1 and 2. Table 3 gives the number and type of generators in each instance.

5.1 Modified Orbital Branching

Because all generators of type k are identical, the set $O^k_{t,1} = \{u^k_{t,1},\ u^k_{t,2}, \ldots,\ u^k_{t,n_k}\}$ is an orbit with respect to $\mathcal{G}$ for all t in T. Moreover, $\texttt{Proj}_{O(k,t)}(\mathcal{G}) \cong S^{n_k}$ so modified orbital branching can be applied when orbit $O^k_{t,1}$ is chosen for branching.

We report results for the following algorithms:

- **Default CPLEX:** CPLEX's default algorithm except that multithreading is turned off.
- **Branch & Cut:** CPLEX with advanced features turned off (mimicking what happens when callbacks are used); CPLEX's symmetry-breaking procedure is used.

Table 1 Generator Data

Gen	$\overline{P}$ (MW)	$\underline{P}$ (MW)	UT (DT) (h)	SU (SD) (MW)	RU (RD) (MW/h)	t^{cold} (h)
1	455	150	8	150	225	5
2	455	150	8	150	225	5
3	130	20	5	20	50	4
4	130	20	5	20	50	4
5	162	25	6	25	60	4
6	80	20	3	20	60	4
7	85	25	3	25	60	2
8	55	10	1	20	135	0

Table 2 Generator Costs

Gen	$CostLow$ ($/MW)	$CostHigh$ ($/MW)	hc ($)	cc ($)
1	16.3	16.6	4500	9000
2	17.4	17.55	5000	10000
3	16.7	17	550	1100
4	16.6	16.95	560	1120
5	19.9	20.75	900	1800
6	22.5	23.3	170	340
7	27.8	27.85	260	520
8	27	27.25	30	60

Table 3 Number of Generators

	Total	Generator							
Instance	Gens	1	2	3	4	5	6	7	8
1	46	10	11	9	1	4	4	2	5
2	46	13	12	1	2	3	8	7	0
3	47	16	19	0	2	0	1	3	6
4	47	11	18	0	8	0	3	5	2
5	48	16	14	2	9	1	3	2	1
6	49	13	14	7	5	3	5	1	1
7	49	10	18	6	1	8	1	3	2
8	49	15	15	0	2	6	2	0	9
9	52	13	10	7	5	6	5	4	2
10	52	16	15	8	0	5	0	2	6
11	52	11	13	4	6	3	7	5	3
12	52	15	14	5	7	0	3	0	8
13	53	12	14	4	4	4	2	4	9
14	55	11	18	7	9	6	1	2	1
15	55	11	15	2	5	8	5	6	3
16	56	10	14	4	8	3	5	9	3
17	56	13	19	6	9	3	1	2	3
18	57	13	16	7	5	3	5	6	2
19	58	14	13	7	2	9	3	7	3
20	62	10	19	8	5	1	8	4	7
21	63	18	10	5	7	6	8	7	2
22	63	19	16	8	2	7	5	1	5
23	67	15	18	8	5	8	7	5	1
24	67	17	19	9	1	5	3	6	7
25	72	15	17	9	7	9	9	5	1

- **OB:** Original orbital branching implemented using callbacks.
- **Modified OB:** Modified orbital branching from Section 3 implemented using callbacks.
- **Modified OB RMRI:** Modified orbital branching implemented using callbacks with the RMRI branching rule to guarantee only non-isomorphic solutions are explored.

The computational results are reported in Table 4. Instances not solved within 2 hours are denoted by "-".

All versions of orbital branching were implemented using the branch callback feature. Branching decisions in the default version were determined by CPLEX. After CPLEX chooses a branching variable, OB computes the orbit of that variable, and branches on that orbit. Modified OB is done in the same way. Modified OB RMRI needs to tests if the orbit is branchable or not. If the orbit is not branchable, then a branchable orbit is found by using the techniques in Section 4.4.

Unfortunately, using callback functions disables other CPLEX features, notably dynamic search. For this reason, in addition to comparing classical orbital branching with the versions of modified orbital branching, we also give results based on CPLEX's default setting (dynamic search plus additional features) and CPLEX with features disabled (using traditional branch-and-cut).

Amongst the three variants of orbital branching, the one with the RMRI branching rule seems to be the most effective, indicating that for these instances, the ability to remove all isomorphic solutions from the search outweighs the restriction in branching. The difference between using RMRI and modified orbital branching without any branching rule, however, isn't nearly as significant as the difference between modified orbital branching and standard orbital branching.

These tests also show the extreme difference between CPLEX with its advanced features and CPLEX using callback functions. One explanation for this extreme difference may be in how CPLEX handles symmetry. Perhaps CPLEX uses more sophisticated strategies that are turned off when callback functions are used. Given the large discrepancy, we wonder how modified orbital branching (even without the RMRI rule) would perform if it were integrated into CPLEX's advanced features, or if CPLEX's advanced symmetry handling is better than modified branching.

In order to better understand the impact of modified orbital branching over traditional orbital branching, consider the partial solution

$$x = \begin{pmatrix} 1 & ? & ? & ? & ? \\ ? & ? & ? & 1 & ? \\ ? & 1 & ? & ? & ? \\ ? & 1 & 1 & ? & ? \end{pmatrix}.$$

Instance	Time					Nodes				
Number	CPLEX	Branch & Cut	OB	Mod OB	RMRI	CPLEX	Branch & Cut	OB	Mod OB	RMRI
1	53.32	2273.09	563.42	169.38	147.09	502	73414	19060	3253	2826
2	30.55	39.33	281.37	76.33	122.11	600	310	6652	1148	2277
3	17.82	762.67	140.05	63.82	62.20	140	14868	3851	190	190
4	40.14	2828.53	2167.12	254.76	358.04	482	103751	69818	4130	7041
5	105.91	2041.84	2320.65	225.80	314.91	2940	55110	38590	2723	4026
6	70.90	1376.24	4068.63	1234.85	591.47	491	28903	66959	17890	7826
7	117.50	-	1647.25	390.06	363.72	2965	-	26230	5685	5191
8	62.44	6504.09	607.31	202.35	437.12	634	253700	17982	4191	10020
9	46.87	888.91	1619.49	659.79	328.56	528	25473	35050	6530	2651
10	62.65	2865.00	444.16	173.42	199.00	530	54672	6141	1255	1440
11	72.20	1617.86	306.28	156.69	141.14	926	35972	3294	1759	1525
12	139.62	-	1523.79	405.84	313.90	4065	-	32521	3597	2534
13	88.08	3390.10	1651.86	511.44	344.16	1462	107773	66000	11000	6701
14	244.20	-	5047.17	1409.94	1070.90	5666	-	54911	15342	9785
15	55.54	703.66	606.59	632.51	308.49	537	22024	10722	11280	4490
16	33.45	97.05	57.87	39.21	41.27	473	2202	150	130	130
17	41.44	2283.43	2380.70	71.09	63.33	331	46536	40350	370	319
18	43.09	4329.38	616.90	546.59	245.39	514	121164	14577	8000	2797
19	148.54	-	4220.13	869.52	732.14	1067	-	76105	12000	9826
20	52.19	4220.21	372.81	273.00	278.31	520	113000	7100	4270	5000
21	45.25	6041.21	1353.21	549.14	502.41	499	178100	30650	7489	6544
22	61.11	6547.76	982.28	207.89	214.42	486	147400	15799	1486	1487
23	61.08	6013.76	524.74	415.28	277.52	499	159917	6148	2844	1476
24	54.92	1538.94	161.21	119.44	122.15	521	29972	1332	623	623
25	78.50	2921.97	906.59	376.58	612.73	513	68649	12044	4130	8844
Mean	73.09	3523.40	1382.86	401.39	327.70	1115.64	88378.72	26481.44	5252.60	4222.76
Geo. Mean	61.83	2183.55	830.53	273.78	243.36	729.25	51718.77	14359.09	2946.22	2616.11

Table 4 Computational results on UC instances

In the subproblem, all of the column permutations have been removed from the symmetry group. But suppose that the corresponding LP solution is

$$x_{LP} = \begin{pmatrix} 1 & .95 & 1 & ? & ? \\ ? & ? & ? & 1 & ? \\ .97 & 1 & 1 & ? & ? \\ .95 & 1 & 1 & ? & ? \end{pmatrix}. \tag{28}$$

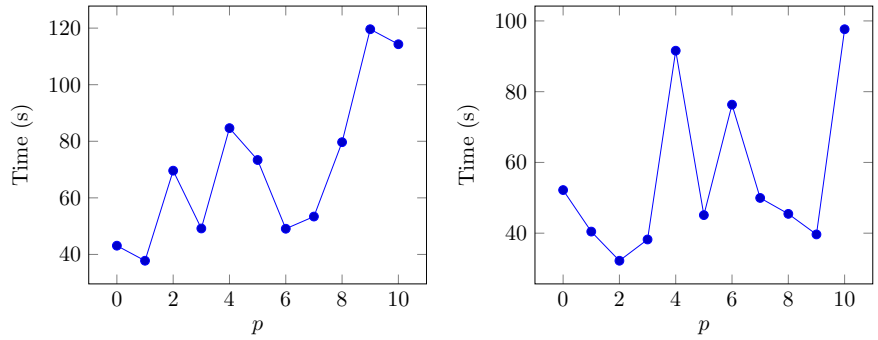
Technically, there is no symmetry found in this subproblem. However, it is likely that the optimal solution to this problem has each of $x_{1,2}$, $x_{3,1}$, and $x_{4,1}$ equal to one. If each of these variables had been fixed, then all permutations of the first three columns of x would be in the subproblem's symmetry group, and those permutations could be used to strengthen the branching disjunction.

We let CPLEX choose our branching candidate, then act on that branching decision by branching on the orbit containing CPLEX's branching candidate. One problem with this approach is that if $x_{i,j}$ is chosen to branch on at a node, then it is unlikely that $x_{i,k}$ will be chosen at a child node. This is especially true in cases similar to (28). Variables that take a value close to one in the LP solution (and especially if they are equal to one) will likely not be chosen for branching because doing so will not improve the bound. However, from a symmetry point of view, those variables need to be fixed to create larger symmetry groups and strengthen the subsequent branches. To exploit symmetry early in the branch-and-bound tree, it is important to branch on variables in the same row as previously fixed variables, but this is not a good branching strategy from a general MILP point of view. Using modified orbital branching makes it more likely that several variables in each row are fixed, so there is a better chance that this symmetry will be recognized.

Interestingly, there is not a great difference between Modified OB and Modified OB RMRI. One possible explanation for this is that the loss of branching flexibility balances out with the full removal of isomorphic solutions. Another explanation might be found in the structure of the UC problem. Because of the minimum up and downtimes, branching on one variable can have a significant effect on several other variables. For instance, if a generator is on at time t then off at time $t + 1$, then the generator must be off for several more time periods to satisfy the minimum downtime constraint.

5.2 Symmetry Breaking Inequalities

The previous section focused on comparing modified orbital branching to other dynamic symmetry-exploiting strategies, such as orbital branching and CPLEX's default strategies to exploit symmetry. Another option for highly symmetric integer programs is to add symmetry-breaking inequalities. The benefit of adding inequalities is that it is much easier to implement as there is no need to modify CPLEX's branching behavior. In this section, the impact of two different sets of symmetry-breaking constraints, Friedman inequalities and column inequalities, is examined.



(a) UC Instance 18 (b) UC Instance 20

Fig. 5 The Impact of p in Friedman Inequalities

As stated in (10), the inequalities are not practical for the UC problem. Because the UC instances are usually solved over a 24-hour time horizon, the Friedman inequalities will have coefficients up to 2^{24}, and solving them may cause numerical difficulties. To address this issue, the Friedman inequalities are relaxed. Rather than enforcing that U^k has lexicographically non-increasing columns, the constraints

$$\sum_{k=1}^{p} 2^{p+1-k} u_{k,j} \geq \sum_{k=1}^{p} 2^{p+1-k} u_{k,j+1} \ \forall j \in \{1, \ldots, \ n-1\}, \tag{29}$$

ensure that the $p \times n_k$ submatrix U_p^k, where $U_p^k = u_{t,j}^k$ for all $1 \leq t \leq p$ and $1 \leq j \leq n_k$, has non-increasing columns.

To test the impact of the relaxed Friedman inequalities, all 25 instances of the UC problem are solved with p varying from 0 (no symmetry-breaking inequalities) to 10. Similarly, the impact of the column inequalities was tested on the same instances. The default CPLEX settings were used (except that the thread count was limited to 1), including symmetry breaking. The results indicate that for the Friedman inequalities, the larger the p, the longer CPLEX takes to solve the problem. Figure 5 illustrates this using UC instances 18 and 20, where the best solution times are usually obtained when p is small. While only two instances are shown for the Friedman inequalities, they are representative of the entire set of instances. Similarly, the inclusion of the column inequalities had a negative impact on the overall solution time. The average solution time using default CPLEX was 73 seconds (a geometric mean of 62 seconds), compared to an average of 116 seconds (with a geometric mean of 102 seconds) after the column inequalities were included. In addition, the size of the branch-and-bound tree grew by a similar rate after the column inequalities were added.

One reason for this behavior is that the inequalities break the symmetry in the original formulation, so CPLEX's symmetry breaking procedures are not used (except for the Friedman inequalities with $p = 0$ that corresponds

to no symmetry-breaking inequalities). Another factor coming into play is that adding the inequalities makes finding feasible solutions more difficult, as many feasible solutions in the original problem become infeasible with the inequalities present. Note however that the symmetry-breaking constraints can be pretty weak, as they do not take integrality into account. There may be classes of tighter symmetry-breaking inequalities that improve performance.

One benefit of using constraints to break symmetry is that doing so does not restrict branching. However, one would expect that adding constraints like (29) would significantly increase the importance of variables with low time indices. For example, fixing $u_{1,i}$ to 0 would also fix $u_{1,j}$ to 0 for $i < j \leq n_k$ (similarly, fixing $u_{1,i}$ to 1 would also fix $u_{1,j}$ to 1 for $1 \leq j < i$). On the other hand, fixing $u_{20,i}$ to either 0 or 1 will likely not be helpful in fixing additional variables (using the Friedman constraints). So, while there is no explicit branching rule, we intuitively argue that these constraints lead to branching that will closely resemble the MI branching rule.

As mentioned in previous sections, the row indices are arbitrary. To test the impact of different row orderings, the symmetry-breaking constraints are modified to reflect different row orderings. If R is a proper ordering of the rows, the permuted Friedman constraints

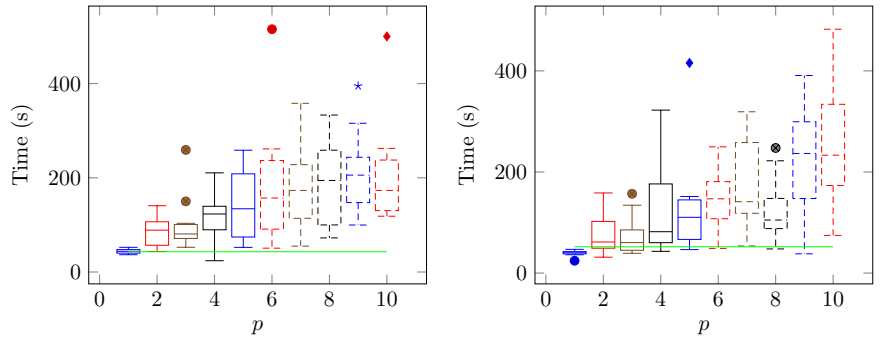
$$\sum_{\{k|R_k \leq p\}} 2^{p+1-R_k} u_{k,j} \geq \sum_{\{k|R_k \leq p\}} 2^{p+1-R_k} u_{k,j+1} \; \forall j \in \{1, \ldots, \; n-1\} \tag{30}$$

ensure that the $p \times n_k$ submatrix U_p^k, where $U_p^k = u^k_{R_t^{-1},j}$ for all $1 \leq t \leq p$ and $1 \leq j \leq n_k$, has non-increasing columns. Let R_i^{-1} denote the index of the row that has rank i. The permuted column inequalities

$$u_{i,j} \leq \sum_{k=1}^{R_i} u_{R_k^{-1},j-1} \; \forall i \in 1, \; \ldots, m, \; \forall j \in 2, \; \ldots, \; n, \tag{31}$$

also ensure that U_p^k has non-increasing columns.

Ten random ordering vectors were generated for each of the UC instances and used to seed both Friedman and column symmetry-breaking inequalities. The Friedman inequalities were tested for values of p from 0 to 10. While the tests were performed on all 25 instances, only data from instances 18 and 20 using Friedman inequalities are reported in Figure 6. For reference, a line showing the solution time without any Friedman inequalities is added to the figures. Interestingly, the data shows that there is a large variance in solution times when varying the row ranks. However, even the best times with the Friedman inequalities are frequently worse than, or not significantly better than, CPLEX without the symmetry-breaking constraints. The column inequalities fare a little better. Figure 7 shows the distribution of the solution times for 10 random row-rankings for each of the 25 UC instances. Again, a line is included to show CPLEX's default solution time. In some cases (for example, instances 12 and 14), adding the column inequalities consistently produces a notable computational speedup, but more often than not, the best



(a) UC Instance 18 (b) UC Instance 20

Fig. 6 The Impact of p in Random Friedman Inequalities

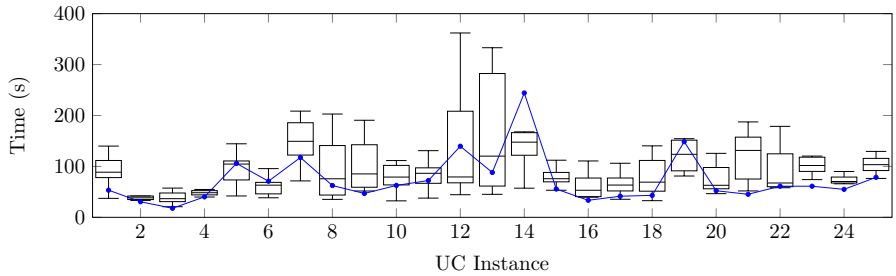


Fig. 7 The Impact of Column Inequalities on Solution Times

time with the column inequalities is only marginally better than CPLEX's default.

While symmetry-breaking constraints are the easiest to implement (as their implentation does not require the use of callback functions), the computational experiments performed in this section seem to indicate that symmetry-breaking constraints are not likely to offer a significant improvement over general symmetry breaking techniques, such at those used by CPLEX and orbital branching.

6 Conclusion

There are generally two different ways to address symmetry in integer programming problems, either to add symmetry-breaking constraints, or to exploit symmetry through branching decisions. This paper takes the latter approach by exploring how the specific structure of the symmetry group of a MILP can be used to strengthen orbital branching. This strengthening, called modified orbital branching, can have a considerable impact on the overall time needed to obtain a global optimal solution. This is demonstrated by testing modified orbital branching on several instances of the unit commitment problem, where

modified orbital branching is able to solve the UC instances in one-third of the time taken by classical orbital branching. In addition, two types of symmetry breaking constraints, Friedman and column inequalities, are examined and compared to CPLEX. The results indicate that CPLEX's default branching is as effective in dealing with symmetry as the constraints.

Acknowledgements The authors thank an anonymous referee for the constructive reports that helped greatly improve this paper.

The research of the first author was supported by NSF CMMI grant 1332662. The research of the second and third authors was partially supported by NSERC, the Natural Sciences and Engineering Research Council of Canada.

References

1. Anjos, M.F.: Recent progress in modeling unit commitment problems. In: L. Zuluaga, T. Terlaky (eds.) Modeling and Optimization: Theory and Applications: Selected Contributions from the MOPTA 2012 Conference, *Springer Proceedings in Mathematics & Statistics*, vol. 84. Springer (2013)
2. Arroyo, J.M., Conejo, A.J.: Optimal response of a thermal unit to an electricity spot market. IEEE Transactions on Power Systems **15**(3), 1098–1104 (2000)
3. Carrion, M., Arroyo, J.: A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. IEEE Transactions on Power Systems **21**(3), 1371 –1378 (2006)
4. Chang, G., Tsai, Y., Lai, C.: A practical mixed integer linear programming based approach for unit commitment. PES General Meeting pp. 221 – 225 (2004)
5. Frangioni, A., Gentile, C.: Perspective cuts for a class of convex 0-1 mixed integer programs. Mathematical Programming **106** (2006)
6. Frangioni, A., Gentile, C., Lacalandra, F.: Tighter approximated milp formulations for unit commitment problems. IEEE Transactions on Power Systems **24**(1) (200)
7. Friedman, E.J.: Fundamental domains for integer programs with symmetries. In: Dress, A.W.M., Xu, Y., Zhu, B. (eds.) Combinatorial Optimization and Applications, *Lecture Notes in Computer Science*, vol. 4616, pp. 146–153. Springer (2007)
8. Gattermann, K., Parrilo, P.: Symmetry groups, semidefinite programs, and sums of squares. Journal of Pure and Applied Algebra **192**, 95–128 (2004)
9. Jeroslow, R.: Trivial integer programs unsolvable by branch-and-bound. Mathematical Programming **6**, 105–109 (1974)
10. Kaibel, V., Loos, A.: Branched polyhedral systems. In: IPCO 2010: The Fourteenth Conference on Integer Programming and Combinatorial Optimization, *Lecture Notes in Computer Science*, vol. 6080, pp. 177–190. Springer (2010)
11. Kaibel, V., Peinhardt, M., Pfetsch, M.E.: Orbitopal fixing. Discrete Optimization **8**(4), 595 – 610 (2011)
12. Kaibel, V., Pfetsch, M.: Packing and partitioning orbitopes. Mathematical Programming **114**, 1–36 (2008)
13. Liu, C., Shahidehpour, M., Li, Z., Fotuhi-Firuzabad, M.: Component and mode models for the short-term scheduling of combined-cycle units. IEEE Transactions on Power Systems **24**(2), 976 –990 (2009)
14. Margot, F.: Pruning by isomorphism in branch-and-cut. Mathematical Programming **94**, 71–90 (2002)
15. Margot, F.: Exploiting orbits in symmetric ILP. Mathematical Programming, Series B **98**, 3–21 (2003)
16. Margot, F.: Symmetry in integer linear programming. In: Jünger, M., Liebling, T.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A. (eds.) 50 Years of Integer Programming 1958-2008, pp. 647–686. Springer Berlin Heidelberg (2010)

17. Ostrowski, J., Anjos, M.F., Vannelli, A.: Tight mixed integer linear programming formulations for the unit commitment problem. IEEE Transactions on Power Systems **27**(1), 39 –46 (2012)
18. Ostrowski, J., Linderoth, J., Rossi, F., Smriglio, S.: Orbital branching. Mathematical Programming **126**(1), 147–178 (2009)
19. Rajan, D., Takriti, S.: Minimum up/down polytopes of the unit commitment problem with start-up costs. Tech. rep., IBM Research Report (2005)
20. Sherali, H.D., Smith, J.C.: Improving zero-one model representations via symmetry considerations. Management Science **47**(10), 1396–1407 (2001)
21. Zhai, Q., Guan, X., Cui, J.: Unit commitment with identical units successive subproblem solving method based on Lagrangian relaxation. IEEE Transactions on Power Systems **17**(4), 1250 – 1257 (2002)